ARMY RESEARCH LABORATORY

Development of Simulation Software for NPT-MD

by Andrew Scott

ARL-CR-554 **November 2004**

prepared by

Department of Electrical Engineering
Alabama A&M University
P.O. Box 1146
Normal, AL 35762

under contract

DAAD17-01-P-1238

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5066

ARL-CR-554 **November 2004**

Development of Simulation Software for NPT-MD

Andrew Scott

prepared by

Department of Electrical Engineering
Alabama A&M University
P.O. Box 1146
Normal, AL 35762

under contract

DAAD17-01-P-1238

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2004	Final	1 January 2001–31 January 2002

4. TITLE AND SUBTITLE
Development of Simulation Software for NPT-MD

5a. CONTRACT NUMBER
DAAD17-01-P-1238

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Andrew Scott*

5d. PROJECT NUMBER
CHSSI CCM5

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical Engineering
Alabama A&M University
P.O. Box 1146
Normal, AL 35762

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-WM-BD
Aberdeen Proving Ground, MD 21005-5066

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
ARL-CR-554

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
*Alabama A&M University, Normal, AL.

14. ABSTRACT

This report details the development of simulation software for the Common High Performance Computing Software Support Initiative Project: Visual eXtensible Molecular Dynamics. The software was designed for use in the NPT Ensemble of the Message Passing Interface-based program "Novice-MD." NPT assumes a fixed number of particles, N, constant pressure, P, and temperature, T. Thus, the corresponding change in volume due to system behavior causes the simulation cells to change shape. From a computational perspective, however, the external boundaries of simulation cells (two- and three-dimensional) must remain mutually perpendicular to maintain numerical stability. Algorithms were developed to convert strained, shape changed, simulation cells into equivalent "rectangularized" cells at each time-step in the dynamic simulation, for both Runge-Kutta and Leap-Frog-Verlet integrators. The conversions at each step are tracked throughout the simulation, thereby enabling conversion to the original coordinate system at any intermediate step, and at the conclusion of the simulation. The software was written in Fortran 90 (f90), and interwoven as necessary with the existing Novice-MD code. The capabilities were successfully implemented and tested on the SGI Origin system at the U.S. Army Research Laboratory Major Shared Resource Center. Additionally, the Novice-MD code was ported and tested on a CRAY SV-1 vector platform.

15. SUBJECT TERMS
NPT-MD, molecular dynamics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Betsy Rice
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	20	19b. TELEPHONE NUMBER (*Include area code*) (410) 306-1904

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

INTENTIONALLY LEFT BLANK.

1. Introduction

This document serves as the final written report for agreement DAAD17-01-P-1238 between the U.S. Army Research Laboratory (ARL) and the Alabama A&M University Research Institute. It documents adherence to the following.

1.1 Statement of Work

Development of the Message Passing Interface -based Visual eXtensible Molecular Dynamics (VXMD) compatible modules for the creation, maintenance, and migration of parallel atomic bond lists and the force modules that use the bond lists.

The principal investigator is Dr. Andrew Scott.

The work under this agreement will consist of extending and improving, and parallelizing the serial bond list software written by Dr. Betsy Rice. The extended software will have two components, one for handling the atomic bond lists within the domain decomposition used by VXMD, and the second will be the force and integration routines that use these lists. These lists require the development of a mapping scheme that will transform "skewed" cells into rectangular for two-dimensional (2-D) and three-dimensional (3-D) cases.

The specific deliverables for this task will be a robust set of modules that allow the implementation of a MD simulator assuming the NPT ensemble.

All results and codes will be shared with ARL. The final document will describe algorithm and code development, as well as specifics for interfacing with existing software under CCM-5. Any follow-on computational effort will focus on extending the VXMD. Other issues will be addressed as they arise.

1.2 Period of Performance

This report describes work that was conducted from 1 October 2001–31 January 2002. Preliminary results were provided to ARL in the November 2001 timeframe.

Specific deliverables:

- Routines that transform non-rectangular prism simulation cells to rectangular prism cells,
- Within shape-changing NPT MD simulation, coordinate transforms such that optimal rectangular prism simulation shape is maintained, incorporated into the MPI version of the MD code, and

- For post-simulation analysis, routines for inverse transformation of simulation cell from rectangular prism shape.

This report provides an overview of the algorithms developed and the software produced, and how they were incorporated in the existing MD software. The actual source code was submitted to Dr. Betsy Rice at ARL on 30 January 2002. The new code that was generated, and the existing Novice-MD routines that were modified, are detailed in section 3.*

2. Algorithm Overview

The following describes a methodology for converting the global Cartesian lattice system to a "wrapped" local cell coordinate system.

Given:

- Lattice vectors *[a,b,c]* in global Cartesian format.

Determine:

- Coordinate transformation matrix, *A*, to map the atoms and lattice vectors into the local simulation environment for the initial lattice and at each time step.
- "Wrap" the resultant local lattice vectors into rectangular (right angled) unit cells for solution by the MD code for the initial lattice and at each time step.
- Track the global transformation, A_{global}, to provide conversion back to the initial global coordinates at the end of the simulation, and/or at intervals throughout.

Products:

- The first lattice vector, *a*, must be oriented in the local *x*-direction.
- The next lattice vector, *b* (considering only the components orthogonal to the first lattice vector) must be mapped into the local *y*-direction.
- The third lattice vector, *c* (considering only the components orthogonal to the previous lattice vectors) must be mapped into the local z-direction.

Procedure:

1. Read in lattice vectors *[a,b,c]*.
2. Find a set of global orthogonal vectors *[la,lb,lc]* from *[a,b,c]*.

*For further information regarding dissemination of the software, contact Dr. Betsy Rice, USARL, ATTN: AMSRD-WM-BD, Aberdeen Proving Ground, MD 21005-5066, <betsyr@arl.army.mil>.

(a) Denote first cell vector *l1*.

(1) Global coordinates of *l1* match the global coordinates of the first lattice vector, *a*.

(2) Determine the global unit vector of *l1*, *l1unit*.

(b) Determine the next cell vector (orthogonal components to *l1*) from the lattice vector, *b*.

(1) Find the magnitude of the components parallel to *l1*; denote as *b_shadow,* e.g., *b_shadow = dot_product(b,l1unit).*

(2) Find the components of *b* that are parallel to *l1*; denote as *b_par,* e.g., *b_par = b_shadow*l1unit*.

(3) Find the components of *b* that are perpendicular to *l1*; denote as *b_orth,* e.g., *b_orth = b - b_par=l2*.

(4) Note that the global coordinates of *l2* represent only the components of the original lattice vector, *b,* that are orthogonal to *l1*.

(5) Determine the global unit vector of *l2*, *l2unit.*

(c) Determine *l3,* from the remaining lattice vector, *c*.

(1) Find the components of *c* that are parallel to *l1*, as above. Denote *c_par1*.

(2) Similarly, find the components of *c* that are parallel to *l2* as above. Denote *c_par2*.

(3) Determine the components of *c* that are perpendicular to both *l1* and *l2*. Denote *l3 = c – c_par1 – c_par2*.

(d) *[l1,l2,l3]* are orthogonal cell vectors in global coordinates formed by projection of the initial lattice vectors.

3. Find a set of local cell vectors $[l1_{local}, l2local, l3local]$ from the global vectors *[l1, l2, l3]*.

(a) Find the magnitudes (lengths) of the vectors $|l1|$, $|l2|$, $|l3|$.

(b) Map the magnitude of the *l1* vector to the local *x*-direction.

(c) Map the magnitude of the *l2* vector to the local *y*-direction.

(d) Map the magnitude of the *l3* vector to the local *z*-direction.

(e) For example, the mapping from the global *[l1,l2,l3]* to the local system $[l1_{local},l2_{local},l3_{local}]$ takes the following form:

$$\begin{bmatrix} l1_x & l2_x & l3_x \\ l1_y & l2_y & l3_y \\ l1_z & l2_z & l3_z \end{bmatrix} \Rightarrow \begin{bmatrix} |l1| & 0 & 0 \\ 0 & |l2| & 0 \\ 0 & 0 & |l3| \end{bmatrix}.$$

4. Find the coordinate transformation matrix, *A.*

(a) Denote the set of global vectors *[l1,l2,l3]* as the matrix *L* and the set of local vectors *[$l1_{local}$,$l2_{local}$,$l3_{local}$]* as the matrix L_{local}.

(b) The mapping function takes the form of the equation, $AL = L_{local}$, where A is an unknown coordinate transformation matrix providing the mapping from the global to the local system.

5. Solve for *A*, with the following:

$$A = L_{local} L^{-1}. \qquad (1)$$

6. Convert global coordinates to local.

(a) Determine the local lattice vectors, with $A*[a,b,c] = [a_{local},b_{local},c_{local}]$.

(b) Determine the local atom positions with $A*r_{global} = r_{local}$.

(c) Etc….

7. Determine the form of the "wrapped" rectangular unit cell.

(a) The global matrix *L* and the local matrix L_{local} meet the criteria for the "wrapped" rectangular unit cell vectors since they are orthogonal and created from projections of the initial lattice vectors.

(b) This can be verified by checking that the volume of the "wrapped" cell is equal to that of the original lattice cell.

(c) Atom coordinates can be wrapped to fit into the rectangular unit cell by utilizing the assumption of periodic conditions in the overall lattice structure.

8. Update the historical transformation matrix, which can be used to convert back to the original or "global" coordinate system when required.

(a) If *t=t(0)*, then $A_{history} = A$.

(b) Else $A_{history} = A * A_{history}$.

In the case of NPT/ShapeChange, there is a strain tensor acting on the cell vectors at each iterative step in the simulation. The actual lattice vectors need to be updated with this incremental strain in order to identify specific boundary conditions. The lattice vectors in local coordinates are updated as follows:

- Determine the incremental strain in the simulation cell, *dStrainTensor,* by multiplying the strained local cell vectors, *L_local_strained,* by the inverse of the original local cell vectors, L_local_n,

$$\text{dStrainTensor} = (\text{ L\_local\_strained})*(\text{L\_Local}_n)^{-1}. \quad (2)$$

- Determine the new (strained) actual lattice vectors in local coordinates by multiplying the current lattice vectors by the newly acquired *dStrainTensor*:

$$\text{latticeVectors}_{n+1} = \text{dStrainTensor}*\text{latticeVectors}_n. \quad (3)$$

In the case of *RungeKuttaNPT,* a shape change is required at each of the four steps in the integration method. Therefore, the transformation is applied to each of the relevant variables in each step and any corresponding variables obtained in previous steps. This is required since the Runge-Kutta method adds terms from each of the four steps, so they must be represented in the same coordinate system to provide meaningful results.

In the case of *LeapFrogVerletNPT,* a shape change is required only once in the integration method. It applies to any active variables, as well as historical values that are saved for use in subsequent iterations.

3. Code Overview

The following describes new code that was written and outlines the changes that were made in the existing software to accommodate the new functionality. The description is listed by individual source code file and routine name as appropriate. The Fortran source files are denoted with the .f90 suffix. Individual routines and/or data objects that are being described are listed in italics. Items in single quotes refer to variables and routines within the Novice-MD software. Items in double quotes refer to system variables and routines. The double colon convention is characteristic of software specification. As previously mentioned, contact Dr. Betsy Rice for further information regarding dissemination of the software.

accumulateMod.f90

- *IncrementalOutput::* rotate incremental stress tensor to global coordinates. Call 'IncrementAccumulator'. Rotate 'stress tensor' back to local coordinates.

boundaryConditionsMod.f90

- *BoundaryConditionsUpdate::* if 'ShapeChange', use the 'current_lattice_ vectors' from the transformCell. Otherwise use the simulationCell lattice vectors.

brennerMod.f90

- *BrennerInitialize::* Fixed bug, initialize ‘dimension’ variable.

controlDescMod.f90

- *ControlDescCreate::* changed “rand” and “srand” functions to “ranf” and “seed” functions for use on the CRAY platform. SGI requires the original file.

latticeMod.f90

- *SimulationCellRestart::* if ‘ShapeChange’ then read the transformCell from the restart file with routine ‘SimulationTransformRead’.
- *LatticeNewRead::* if the root process, rotate the atoms into the local system using ‘UtilityRotateCoordinates’, then wrap them into the local simulationCell using ‘UtilityWrapCoordinates’.
- *LatticeWrite::* if’ShapeChange’, output the transformCell to the restart file using ‘SimulationTransfromWrite’.
- *LatticeAtomsPrint::* if the root process, then rotate the atom coordinates into global space, write the data out, then rotate the atom coordinates back into local space using ‘UtilityRotateCoordinates’.
- *Fcc* and *cubic::* changed ‘rand’ to ‘ranf’ for use on the CRAY; reverse for SGI.

moverMod.f90

- *LeapFrogVerletNPT::* is ‘ShapeChange’, incorporate transformCell to allow shape change of the current cell and subsequent cell rectangularization for each iteration, using ‘SimulationCellUpdate’. Rotate coordinate variables; *x, vx, dvx, vold* into new local coordinates using ‘UtilityRotateCoordinates’. Rotate the strain matrix, *eta,* into the new system.

noviceMD.f90 (main program)

- Create the ‘transformCell’ from the initial ‘simulationCell’ input using ‘SimulationCellTransform’.
- Set *ShapeChangeFlag = 1* if ‘ShapeChange’.
- Added ‘DEBUG’ statements to print transformCell.

rungeKuttaMod.f90

- *RungeKuttaNPT::* if ‘ShapeChange’, incorporate ‘transformCell’ to allow shape change of unit cell and subsequent cell rectangularization for each of the four steps in the R-K integration using ‘SimulationCellUpdate’. Rotate coordinate variables; *x, vx, dvx, k0, k1,*

l0, l1, q, and *qv* for each step using ‘UtilityRotateCoordinates’. Rotate the matrices; *eta, e0, e1, h0, h1* into new local coordinate system.

simulationCellMod.f90

- *SimulationCellT::* added a pointer, ‘transformCell’ of type ‘transformCellT’ to the data structure for ready access to the transformation data.
- *SimulationCellCreate::* initialize ‘transformCell’ pointer to NULL.
- *SimulationCellUPdate::* added this routine to update the transform and simulation cell structures during NPT ‘ShapeChange’ procedure.
- *SimulationCell Transform::* added this routine to transform the initial simulationCell lattice vectors to the rectangularized cell for simulation.
- *SimulationCellFormattedWrite::* if ‘ShapeChange’ then transform current lattice vectors to global coordinates. Output global lattice vectors. Transform lattice vectors back into local coordinate system.
- *SimulationTransformWrite::* added this routine to write the transformCell structure to the ‘RESTART’ file, in the case of ‘ShapeChange’.
- *SimulationTransformRead::* added this routine to read the transformCell structure from the ‘RESTART’ file.
- *SimulationCellDestroy::* dispose of ‘transformCell’ structure.
- *CalculateNType::* changed the function for random number generation to RANF for CRAY port, and back to RAND for SGI.

transformCellMod.f90

- *TransformCellT::* added the data structure containing the components needed to transform the ‘simulationCell’ to rectangularized cells. Includes rotational history data and ‘strainTensor’ history information.
- *TransformCellCreate::* new routine to allocate memory and initialize the ‘transformCell’ data structure.
- *TransformCellUpdate::* new routine to update the ‘transformCell’ structure given a new lattice vector configuration.
- *TransformCellProcess::* Performs the actual calculations to find the rotated, wrapped cell and the current transform matrix A.
- *TransformCellPrint::* new routine to release the memory associated with the TransformCell structure.

utilityMod.f90

- *UtilityRealPrint::* new routine to output a formatted array of real numbers. Typically some sort of coordinate information.
- *UtilityWrapCoordinates::* new routine to "wrap" an array of atom coordinates to fit inside a given simulation cell.
- *UtilityRotateCoordinates::* new routine to rotate an array of atom coordinates with a given rotation transform matrix, A.
- *UtilityInverse::* new routine to "wrap" an array of atom coordinates to fit inside a given simulation cell.
- *UtilityMag::* new function to calculate the magnitude of a given vector.
- *UtilityUnit::* new routine to determine the unit vector associated with a given vector
- *UtilityPar::* new routine to determine the parallel components of one given vector to another.
- *UtilityPerp::* new routine to determine the perpendicular components of one given vector to another.
- *UtilityCross_3d::* new routine to determine the vector cross product of two 3-D vectors.
- *UtilityGauss::* Public domain f77 LINPACK LU decomposition routine, which was modified with f90 syntax and customized for use with this project.
- *UtilityGedi::* Public domain f77 LINPACK routine, which inverts the LU decomposed matrix resultant of UtilityGauss and calculates its determinant (used for cell volume calculation). It was modified with f90 syntax for this project.
- *UtilityScal::* Public domain f77 BLAS routine that scales a given vector by a constant. It was modified with f90 syntax for this project.
- *UtilitySaxpy::* Public domain f77 BLAS routine that calculates a constant times a vector plus a vector. It was modified with f90 syntax for this work.
- *UtilitySwap::* Public domain f77 BLAS routine that interchanges two vectors. It was modified with f90 syntax for this work.
- *UtilityIsamax::* Public domain f77 BLAS routine that returns the index number of the element having the maximum value in a given real array. It was modified with f90 for use in this project.

initialVelocitiesMod.f90

- *Gauss*:: changed random number generator to “RANF” for use on the CRAY and back to “RAND” for the SGI.

mallopt.c

- *MALLOPTDEBUG/MALLOPTNODEBUG*:: added an extern “C” declaration for use on the CRAY. Changed the ‘M_DEBUG’ keyword to ‘M_TRACE’ on the CRAY and back for SGI. The keywords differ between the platforms.

putAttributeMod.f90, conserveMod.f90, ewaldMod.f90, potentialMod.f90, radial.f90, radialMod.f90, rungeKuttaMod.f90, workMod.f90

- Changed the “double precision” variable type to “real (double)” for consistency and porting to the CRAY.

makefile (/Cells directory)

- Modified the ‘makefile’ to include the new source code files, transformCellMod.f90 and utilityMod.f90 for inclusion in the compilation process.
- Updated the dependency map to ensure appropriate ordering.

4. Verification

The methodology and concomitant code were tested and verified using one-dimensional, 2-D, and 3-D cell vectors. The matrix inversion and vector routines are dimensionally independent, with the exception of “UtilityCross-3d”, which requires 3-D vector input.

The updated program was compiled and tested on CRAY SV1 and SGI Origin2000 platforms. The preliminary tests indicate that the desired ‘NPT/ShapeChange’ functionality is observed.

5. Deliverables

The source code for both CRAY and SGI versions were electronically submitted to Dr. Rice on 30 January 2002. This report satisfies the written documentation necessary for completion of the project. As mentioned previously, Dr. Rice may be contacted for information regarding dissemination of the software.

6. Future Work

Subject to the availability of continued funds, future work will include updating the cell's neighbor lists and incorporating parallel force communications to the MD simulations, or any other work deemed necessary by Dr. Rice.

7. Conclusions

This report has outlined the conformance and production of the project deliverables, specifically:

- Routines that transform non-rectangular prism simulation cells to rectangular prism cells were created and tested.
- Within shape-changing NPT MD simulation, coordinate transformations to maintain the optimal rectangular prism simulation shape were developed and introduced into the MPI version of MD code.
- For post-simulation analysis, routines for inverse transformation of simulation cell from rectangular prism shape were created.

In addition, the MD code was ported and tested on the Cray SV-1 vector platform.

NO. OF COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD
SYSTEMS OF SYSTEMS INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 US MILITARY ACADEMY
MATH SCI CTR EXCELLENCE
MADN MATH
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE AD IM DR
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

NO. OF COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

1 ALABAMA A&M
DEPT ELEC ENGNRNG
A SCOTT
PO BOX 1146
NORMAL AL 35762

ABERDEEN PROVING GROUND

25 DIR USARL
AMSRL WM BD
W R ANDERSON
R A BEYER
A L BRANT
S W BUNTE
C F CHABALOWSKI
T P COFFEE
J COLBURN
P J CONROY
B E FORCH
B E HOMAN
S L HOWARD
P J KASTE
A J KOTLAR
C LEVERITT
K L MCNESBY
M MCQUAID
A W MIZIOLEK
J B MORRIS
J A NEWBERRY
M J NUSCA
R A PESCE-RODRIGUEZ
G P REEVES
B M RICE
R C SAUSA
A W WILLIAMS